Resources

http://slides.com/thierrydelprat/at-overview

Document blobs

- https://doc.nuxeo.com/nxdoc/file-storage/
- https://doc.nuxeo.com/nxdoc/java-client/#repository-api
- http://community.nuxeo.com/api/nuxeo/latest/javadoc/org/nuxeo/ecm/core/blob/DefaultBlobDispatcher.html
- http://explorer.nuxeo.com/nuxeo/site/distribution/Nuxeo%20Platform%20LTS%202019-10.10/viewExtensionPoint/org.nuxeo.ecm.core.blob.BlobManager--configuration

Document Metadata

- https://doc.nuxeo.com/nxdoc/vcs/
- https://doc.nuxeo.com/nxdoc/dbs/

Indexes

- https://doc.nuxeo.com/nxdoc/indexing-and-query/
- https://doc.nuxeo.com/nxdoc/elasticsearch-indexing-logic/
- https://doc.nuxeo.com/nxdoc/configuring-the-elasticsearch-mapping/
- https://doc.nuxeo.com/nxdoc/search-endpoints/

Directories

- https://doc.nuxeo.com/nxdoc/data-lists-and-directories/
- http://community.nuxeo.com/api/nuxeo/latest/javadoc/org/nuxeo/ecm/directory/Directory.html#getSession--
- http://community.nuxeo.com/api/nuxeo/latest/javadoc/org/nuxeo/ecm/directory/api/DirectoryService.html
- https://nuxeo.github.io/api-playground/#/resources/directory/0/GET
- http://explorer.nuxeo.com/nuxeo/site/distribution/server-10.10/viewExtensionPoint/org.nuxeo.ecm.directory.sql.SQLDirectoryFactory--directories
- http://explorer.nuxeo.com/nuxeo/site/distribution/server-10.10/viewComponent/org.nuxeo.ecm.directory.sql.SQLDirectoryFactory

Audits

- https://doc.nuxeo.com/nxdoc/audit/

- http://explorer.nuxeo.com/nuxeo/site/distribution/server-10.10/viewService/org.nuxeo.ecm.platform.audit.api.AuditReader
- http://explorer.nuxeo.com/nuxeo/site/distribution/server-10.10/viewService/org.nuxeo.ecm.platform.audit.api.AuditLogger

Streams

- https://doc.nuxeo.com/nxdoc/nuxeo-stream/
- https://github.com/nuxeo/nuxeo/tree/master/nuxeo-runtime/nuxeo-runtime-stream#the-log-configuration
- http://community.nuxeo.com/api/nuxeo/release-10.10/javadoc/org/nuxeo/runtime/stream/StreamService.html
- https://doc.nuxeo.com/nxdoc/kafka/

Key/Value

- https://doc.nuxeo.com/nxdoc/nuxeo-and-redis/
- https://doc.nuxeo.com/nxdoc/redis-configuration/

Counters

- http://explorer.nuxeo.com/nuxeo/site/distribution/server-10.10/viewOperation/Counters.GET

Caches

- https://doc.nuxeo.com/nxdoc/dbs-cache/
- https://explorer.nuxeo.com/nuxeo/site/distribution/server-10.10/viewComponent/org.nuxeo.ecm.core.cache.CacheService

Pub/Sub

Transcript

--- VIDEO #1 ---

Introduction

Welcome to the **Nuxeo Architecture** course! This course covers the basics of Nuxeo Platform architecture. You‘ll learn about its main components, which data Nuxeo Platform handles, and how. You’ll also get a better understanding of the standard Nuxeo Platform cluster set up. In this first video, we’ll look at all the data sources necessary for Nuxeo Platform to function properly.

Architecture Principles

The Nuxeo Platform is based on a configurable and extensible component model throughout all the levels of the platform: UI, Nuxeo Services and Nuxeo Repository. This means that the persistence layers of the Nuxeo Platform repository are configurable in the same way. There are several options depending on scalability, integration, and performance constraints, making the Nuxeo Platform truly cross-platform. This pluggability even allows you to implement custom data backends if needed.

One way to distinguish between data types handled by the Content Platform is persistence. Persistent data should be saved in the repository and can therefore survive a crash. Here, we'll find all the business information. Non-persistent data refers to technical components which are used to ensure the best performance of Nuxeo Platform: cluster management, indexing and so on.

In the next video, we'll start reviewing the persistent components of the Nuxeo Platform, starting with Document blobs. Thanks for watching!

--- VIDEO #2 ---

Welcome back to the Nuxeo Architecture course. In this video, we'll look at the management of Document blobs within Nuxeo Platform.

What is persisted

Document blobs

Document blobs refer to all the binary files imported to Nuxeo Platform and are an important part of the persisted data. They can include PDFs, videos, images and more. These binaries are structured within a binary stream, and the address of where the binary stream is persisted is stored in the database so that it can be retrieved.

The component that persists the binary stream is called a blob provider. The blob provider has an API that allows you to read and write binary streams as well as other useful methods, such as a direct download URL of the content, or different conversions of a document. They can also implement several encryption mechanisms.

There are several components involved in the document creation process:

The Blob Manager is a service that's called when a document is created. As a Nuxeo Platform instance can make use of several Blob Providers on the same instance, we need to identify which Blob Provider should be used: this is the role of the Blob Dispatcher. You can implement easy binary routing, such as redirecting a blob according to its type or mime-type, or more advanced business logic. Once a suitable Blob Provider is identified, it's called by the blob manager. The Blob Manager is a storage driver which physically stores the document in the Blob Storage. The blob storage has features such as content de-duplication: Files from a blob storage are never moved or updated: they are only removed via a GarbageCollection process, unless this is a requirement (such as in an SEC-17-a4 context).

In the next video, we'll see how Nuxeo Platform manages document metadata. Thanks for watching !

--- VIDEO #3 ---

Welcome back to the Nuxeo Architecture course. In this video, we'll present the Document metadata management within Nuxeo Platform.

Document medatada

Document metadata, also known as document properties, are stored in a database which permits queries. There are several kinds of document properties:

- System properties such as document ID, Path, repository, and facets
- Standard or generic properties such as those belonging to the dublincore schema, refer to the document creator, title, creation date and contributors
- Custom properties are those created to answer a specific business need, such as an invoice number, client reference or invoice amount, for example
- Related properties concern all the information surrounding a document: permissions, conversion information, tags, comments and more.

The "DocumentStore" handles the hierarchy and properties of all content. Nuxeo Platform provides two different interfaces for implementing the DocumentStore: Visible Content Storage, or VCS, and Document-Based Storage, or DBS.

Nuxeo VCS was designed for persisting metadata in the major Relational Database Management Systems on the market. It provides a clean SQL Mapping between XSD schemas and relational tables, allowing transparent access to the data at SQL level.

DBS implementation stores Nuxeo documents inside a document-oriented store, like MongoDB. It's designed to maximise read/write throughput and benefit from NoSQL high scalability. Effectively, the MongoDB implementation of the Nuxeo repository tops all benchmarks.

In the next video, we'll see how Nuxeo Platform manages document audit trails. Thanks for watching!

--- VIDEO #4 ---

Welcome back to the Nuxeo Architecture course. In this video, we'll look at audit trail management within Nuxeo Platform.

Audit trail

The audit trail keeps a record of what happens in the Nuxeo repository and can be useful for technical, security, business or legal reasons. The audit service can be accessed directly with the Java API for writing audit entries, but the main source for Audit entries is the Nuxeo event bus, as there is a direct bridge between it and the audit trail, with the ability in the audit service to configure which events should be logged.

The audit service provides several interfaces: auditLogger, AuditReader, AuditAdmin and AuditStorage. It would probably need a bit of refactoring to improve consistency with features added over time. But the important things to remember about the audit service are that:

- it's possible to switch between Elasticsearch and SQL backends
- Elasticsearch is the recommended backend, especially when querying the audit is a common use case (typically when Nuxeo Drive is deployed and used).
- it's possible to deploy a configuration where audit entries are written in two different backends at the same time, for security reasons.

Audit log entries are customizable:

- you add "extended" information to an audit entry, in order to capture more data
- you can filter which entries are logged or not.

In the next video, we'll see how Nuxeo Platform manages the document index. Thanks for watching!

--- VIDEO #5 ---

Welcome back to the Nuxeo Architecture course. In this video, we'll look at index management within Nuxeo Platform.

Indexes

Accessing data through an index versus querying the database directly is a standard pattern in the data world for performance purposes. Nuxeo Platform uses the Elasticsearch engine, which lets you access thousands of documents in just seconds. The integration is done in such a way that allows for real time indexing. There is nearly no latency between document creation and the moment it is appears in the index. Queries relying on Elasticsearch are secure and you can either query with NXQL or by using the native Elasticsearch REST API.

Nuxeo Platform relies on the ElasticSearch index backend for the best performance as it provides:

- queries on terms using inverted index
- very efficient caching
- native full text support & distributed architecture

Elasticsearch offers features such as hints, geolocalization or synonym parameters.

You can also define your own mapping to answer specific use cases, such as exclude field from the full-text search. In this case, you need to create a custom template that redefines the Elasticsearch mapping. This way the mapping reference stays on the Nuxeo configuration side -- you should not update the mapping directly on the Elasticsearch side.

Nuxeo exposes a search API that can be used by all its clients. The platform also exposes a limited set of Readonly Elasticsearch HTTP REST API using Nuxeo authentication and authorization.

In the next video, we'll see how Nuxeo Platform manages directories. Thanks for watching!

--- VIDEO #6 ---

Welcome back to the Nuxeo Architecture course. In this video, we'll look at directory management within Nuxeo Platform.

Directories

Directories provide an abstraction of all referential data that can be manipulated inside an application. Directories are similar to a standard SQL tables. Directories are abstracted in order to provide a generic service so that the API calls are independent from the possible directory implementations. Here are some examples of manipulating directories with the Java API and the REST API.

Directories are used in the following cases:

- Gathering lists of values for vocabularies. In addition to vocabularies created in Nuxeo Studio, you can rely on external data sources, and use the SQL directory implementation that can map SQL tables.
- Managing users and groups. The LDAP directory implementation is used to retrieve users and groups stored in an external corporate user and group directory, such as an Active Directory. Users and groups can come from different directory branches, in a widely-distributed organization, for example. For this case, it's possible to use the Multidirectory implementation which combines values coming from different directories.
- Mapping documents inside the Repository, with the Repository Directory implementation, or a Custom directory relying on Directory Connector to map a remote service

All directories share a common set of properties that can be defined directly inside a specific directory, or through a templating mechanism: a directory can be read-only, it can load the first set of values from a CSV file, it can control how data are created within a directory with the createTablePolicy and even control how entries are cached.

It's very common to write a new implementation of the directories to fetch data from other backends / webservices and fill in UI value suggestions lists.

In the next video, we will see how Nuxeo Platform manages streams. Thanks for watching!

--- VIDEO #7 ---

Welcome back to the Nuxeo Architecture course. In this video, we'll look at stream management within Nuxeo Platform.

Streams

Streams are used to handle asynchronous processing. They record messages in a log and can be accessed in a specific order as many times as needed. Streaming design brings great resilience to the platform in:

- Fault tolerance
- Durability
- Immutability
- And ordering

Streams are used in the platform for:

- Audits (Stream Audit Writer)
- Bulk action framework
- Scheduled works (with Stream WorkManager scaling the number of job requests much better than with Redis, which is memory-constrained)
- Import tasks

Streams will be used for an even wider scope in the future.

It’s shipped with several implementations:

- Kafka is the privileged implementation because it brings more resilience and a greater distribution capability, such as having several different consumer threads for the same queue.
- Chronicle Queue, for the embedded distribution; the queue is written on the file system. It can also be used in a production environment, but it is not fully fault tolerant as you have to wait for the consumers to finish processing before properly shutting down a nuxeo server, otherwise some work might be lost.

The nuxeo-runtime-stream module provides an integration of nuxeo-stream by exposing two services: The Kafka Configuration Service, to register Kafka and Zookeeper access, and the StreamService service which provides:

- A way to register different Log configurations
- An access to a Log manager that can create Log, create tailer (reader) or appender (writer).
- A way to register stream processing.

The Nuxeo stream service is used in different locations.

- The stream audit writer to store The events that need to be traced in the audit

- The StreamWorkmanager to execute work than can be queued without worries about the memory limits.
- The Stream PubSub Provider to handle cluster invalidation.
- The Bulk Action Framework to run resilient bulk actions on documents.

In the next video, we'll look at additional persistent data in Nuxeo Platform. Thanks for watching!

--- VIDEO #8 ---

Welcome back to the Nuxeo Architecture course. In this video, we'll look at additional persistent data within Nuxeo Platform.

Other persistent Data

Key/Value

Key/Value stores are used for several things in the platform, including sharing values across all servers of a cluster, sharing sequences, sharing document states, invalidating caches, storing asynchronous work states and more. The platform uses Redis or MongoDB as a persistence solution.

Counters

Counters are simple incremented sequences. They are generally used to build some property values, and can be easily used in Nuxeo Studio.

Transient Store

A Transient Store allows you to store temporary blobs and associated parameters, such as the file name or MIME type, outside of the document repository.

It is typically used by:

- The Batch Upload API, to temporarily store a batch of uploaded blobs until they are attached to a document.

- The ConversionService to store the BlobHolder resulting from Asynchronous Conversion Work.
- The BatchHandler contributions, which interact with the Transient Store.

In the next video, we'll look at the data that isn't persisted in the Nuxeo Architecture. Thanks for watching!

--- VIDEO #9 ---

Welcome back to the Nuxeo Architecture course. In this video, we'll look at the data that isn't persisted.

What is not persisted

Caches

Cached data can be recomputed from a source. The goal of a cache is to improve the repository efficiency. A database cache exists on DBS repositories such as mongoDB, and is configurable from the nuxeo.conf file: For example, you can set the max number of elements contained in the cache or the Time To Live defines the time before the cache will be destroyed in minutes.

Pub/Sub

The PubSub service allows cross-instance notifications through simple messages sent to topics. It publishes messages to topics and receives messages from topics to which you subscribe. It works in connected mode only and can be customized through the corresponding service.

Thanks for watching!